A Unified Tool For Adaptive Collocation Techniques Applied to Solving Optimal Control Problems

Bethany Kelly
University of Central Florida

A UNIFIED TOOL FOR ADAPTIVE COLLOCATION TECHNIQUES APPLIED TO SOLVING OPTIMAL CONTROL PROBLEMS

by

BETHANY KELLY
B.S. University of West Florida, 2020

A thesis submitted in partial fulfillment of the requirements
for the degree of Master of Science
in the Department of Mechanical and Aerospace Engineering
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Summer Term
2022

ABSTRACT

In this work, a user-friendly MATLAB tool is introduced to solve nonlinear optimal control problems by applying collocation techniques using Coupled Radial Basis Functions (CRBFs). CRBFs are a new class of Radial Basis Functions combined with a conical spline r^5, which provides the advantage of insensitivity to the shape parameter while maintaining accuracy and robustness. To solve optimal control problems, software tools are often employed to implement numerical methods and apply advanced techniques to solving differential equations. Although several commercial software tools exist for solving optimal control problems, such as ICLOCS2, GPOPS, and DIDO, there are no options available that utilize adaptive collocation with CRBFs. A unified MATLAB tool named Radial Optimal Control Software (ROCS) is introduced and not only implements the CRBF method, but also enables any user, from professionals to students, to solve nonlinear optimal control problems through a user-friendly interface. The tool accepts user input for boundary conditions, necessary conditions, and the governing equations of motion. The two-point boundary value problem (TPBVP) is approximated through collocation using CRBFs, and the resulting nonlinear algebraic equations (NAEs) are solved with a MATLAB solver. The tool's usefulness and application are demonstrated by solving classical nonlinear optimal control problems and comparing the results with the solutions found in the literature. Compared to classical numerical method techniques, the present tool is shown to solve optimal control problems more efficiently for the same level of accuracy. By introducing this unified MATLAB tool to solving nonlinear optimal control problems, the intent is to enable professionals and students to solve nonlinear optimal control problems, e.g., in astrodynamics and space-flight mechanics, without the need for extensive manipulation of code in existing software tools and without extensive knowledge of applying numerical solvers.

Dedicated to my Mom and Ne.

ACKNOWLEDGMENTS

I would like to thank Dr. Tarek A. Elgohary and Dr. Ahmed E. Seleit for providing their guidance and expertise during my research and Master's Degree program. This work would not have been possible without their continued support and mentorship.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: BACKGROUND

Classical Control Theory

Control theory is concerned with governing a dynamic system in order to reach a desired goal. Governing the behavior of a dynamic system is typically accomplished through the use of controls [1, 2]. Desirable performance is achieved in terms of optimizing output parameters in the time and frequency domains, such as rise time, overshoot, gain/phase margin, and settling time. When dealing with more complex dynamic systems, such as nonlinear multiple-input multiple-output (MIMO) systems, optimal control theory can be applied to achieve the desired system behavior [3].

Optimal Control Theory

Optimal control theory seeks to find the control that minimizes or maximizes performance criterion, while satisfying any constraints [3, 4]. Optimal control problems consist of algebraic and differential equations that can be linear or nonlinear in nature. Due to the complexity in solving these types of problems, numerical methods are often employed to transcribe the problem into a nonlinear programming problem or a two-point boundary value problem that can be solved. There are two categories of numerical methods that can be applied to solving optimal control problems: indirect methods and direct methods [4, 5].

Indirect Versus Direct Methods

With indirect methods, the calculus of variations and Pontryagin's Maximum Principle are utilized to derive first-order necessary conditions. This results in a two-point boundary-value problem that can be solved numerically to satisfy the boundary conditions [4, 5]. The optimal trajectories generated from indirect methods can be analyzed to determine the optimal solution from the minimum, maximum, and/or saddle point(s) [4]. Advantages of using an indirect method include a high level of confidence in the accuracy in the solution satisfies the first-order conditions for optimality [5]. However, with indirect methods, an initial guess is needed for the costate variables, there is a small radii of convergence, and an a priori knowledge of the constrained arc-sequence is needed if there are path constraints [5, 6].

With direct methods, the optimal control problem is transcribed into a nonlinear programming problem (NLP) that can be solved using different well-known algorithms [4, 5]. Direct methods involve minimizing an objective function, and do not require transversality conditions, control equations, nor adjoint equations like indirect methods do [6]. Direct methods, however, can lack information about the costates [5].

Different methods exist for transcribing an optimal control problem into an NLP, such as the direct shooting method, which involves integrating a system of equations and computing the error in the terminal conditions. However, one drawback of the shooting method is it requires *a priori* knowledge of the system and a good initial guess for convergence [4, 7]. Therefore, for solving optimal control problems, direct collocation methods are often more suitable.

Direct Collocation Methods

Direct collocation methods are considered to be state and control parameterization methods, where an NLP is transcribed by approximating differential equations at collocation points using piecewise polynomials. This involves using a specified functional form for the state and control [4, 5]. The most common types of collocation used are local collocation and global collocation. When employing local collocation, the time interval for the optimal control problem is first divided into subintervals. Within each subinterval, a small amount of collocation points are used and continuity conditions on the independent variable and the state are used to connect the subintervals [4, 8]. The subintervals may also be connected through a continuity condition on the control. Local collocation has long been used to solve optimal control problems because it is simple, efficient, and supports the dynamics' local behavior [6, 8].

Pseudospectral Methods

Although local collocation has historically been used, global collocation methods, which collocate across the time interval using a global polynomial, have risen in popularity. More specifically, a class of direct transcription methods called pseudospectral methods, which utilize the global polynomial for collocation at specified points, have become increasingly popular [4, 8]. Pseudospectral methods and global collocation have proven successful in different applications, to include fluid dynamics and optimal spacecraft formations [9, 10, 11].

The benefit of pseudospectral methods is they converge exponentially, or spectrally, by implementing collocation *orthogonally* using the roots of a global orthogonal polynomial. Unlike local collocation methods, the degree of the polynomial is varied for each subinterval while the number of subintervals, or meshes, is fixed. Chebyshev or Lagrange polynomials typically serve as the basis functions for pseudospectral methods [4, 5, 13, 14, 15], and the most popular types of collocation points are *Legendre-Gauss* nodes, *Legendre-Gauss-Lobatto* nodes, and *Legendre-Gauss-Radau* nodes [4, 8, 12]. For *Legendre-Gauss-Lobatto* nodes, collocation occurs on the derivative at all $M + 1$ approximation points and a global polynomial with degree M is used to approximate the state. This leads to a square and singular differentiation matrix [4, 12, 16]. On the other hand, a non-square and full rank differentiation matrix results from collocation using *Legendre-Gauss-Radau* nodes or *Legendre-Gauss* nodes and collocation does not occur at all approximation points. When these nodes are used, collocation occurs at M points and a global polynomial of degree M is used to approximate the state [4, 12].

Radial Basis Functions

Although pseudospectral methods are a popular choice for solving optimal control problems, one major drawback is their dependency on a specific grid of nodes, which make this method less efficient for interpolating nonsmooth functions [4, 17]. In recent years, researchers have been exploring the use of Radial Basis Functions (RBFs) for solving optimal control problems. With this method, global RBFs are interpolated on an arbitrary set of nodes as collocation points. The arbitrary discretization points are more suitable for approximating nonsmooth functions than pseudospectral methods because they offer flexibility in choosing the collocation points. In addition, there is flexibility in choosing the type of RBF to interpolate and different types can be used for each state and control. The classical RBFs used in solving optimal control problems include Gaussian, multiquadric, inverse quadrics, inverse multiquadric, and polyharmonic splines [17].

RBFs were originally introduced in 1968 by Hardy [18], and first applied to solving partial differential equations in 1990 by Kansa [19]. The applicability of RBFs in solving optimal control problems has been demonstrated in work by Rad et al. [20] and Elgohary et al. [21, 22] with

Legendre-Gauss-Lobatto nodes [20] as well as in work by Mirinejad and Inanc with direct transcription [17]. It is important to note that the convergence and behavior of RBFs in solving optimal control problems is reliant on the shape parameter, which controls the flatness of the function [23]. Therefore, optimizing the shape parameter is desirable to achieve accurate approximations and prevent latency. Research has been conducted by Cheng to investigate how the shape parameter influences the optimal solution [24]. In addition, work by Karageorghis and Tryfonos [25] explored determining the optimal shape parameter by treating it as an unknown along with the RBF coefficients. They determined that the most accurate way to determine the optimal shape parameter was through brute force: determining the sequence of values for the shape parameter and assessing which one provides the best approximation. This was an expensive iterative approach [25].

Coupled Radial Basis Functions

To address the challenges associated with the shape parameter, a new class of RBFs called Coupled Radial Basis Functions (CRBFs) were recently introduced. In work by Zhang, numerical examples are solved by using Kansa's method paired CRBFS, which are classical RBFs that are combined with an r^5 conical spline. This method demonstrated accuracy and robustness while remaining insensitive to the shape parameter [26]. The idea of CRBFs was extended to solving nonlinear optimal control problems through work by Seleit and Elgohary, where Zermelo's problem, the nonlinear duffing oscillator, and the inverted cart-pole problem were solved using CRBFs [27].

This thesis will be focused on solving optimal control problems with a unified tool that utilizes CRBFs to collocate in a manner that provides accurate results while not being reliant on a certain nodal distribution nor interpolation. It builds off the research conducted by Seleit and Elgohary in applying CRBFs to solving nonlinear optimal control problems. The goal is for the tool to provide a user-friendly interface for solving optimal control problems with the CRBF method.

Optimal Control Software Tools

Software tools are often employed to implement numerical methods and solve optimal control problems. Many different software tools have been developed to tackle more complex problems and apply advanced techniques to numerically solve differential equations. These tools started becoming available in the 1980s, and were often based in FORTRAN. However, they were revolutionary in solving more realistic challenges, such as those in aerodynamic trajectory and chemical engineering control [28]. More modern optimal control software tools are based in MATLAB due to the wide availability of numerical solvers and engineering toolboxes. The MATLAB environment enables custom and automatic code generation that can be targeted at specific applications [29, 30].

Most of the available commercial software tools solve optimal control problems via direct methods. However, some softwares such as DIDO and OptimTraj pair indirect methods with NLP solvers [28]. This section will focus on providing a general overview of the most common optimal control software tools available on the market.

ICLOCS2

Imperial College London Optimal Control Software 2 (ICLOCS2) is a comprehensive toolbox that utilizes direct transcription methods such as direct collocation and direct multiple shooting methods to solve optimal control problems in MATLAB or Simulink environments. It succeeds the previous ICLOCS version [31]. ICLOCS2 provides accurate results for classical problems and real-time optimal control-based problems, such as model predictive control (MPC). The benefit of ICLOCS2 is it provides flexibility when it comes to the trade-off between computational power and solution accuracy [32]. In terms of discretization methods, ICLOCS2 offers several options, such as Hermite-Simpson (h methods) and Pseudospectral Methods (p/hp methods) when fast convergence and high accuracy are required. This version also offers additional tools such as closed-loop simulations, automatic meshing, mesh refinement, and automatic scaling [32]. The underlying NLP solvers used are IPOPT, *fmincon*, and WORHP [33, 34, 35]. IPOPT, or Interior Point Optimizer, is an open-source library for solving NLPs, *fmincon* is a MATLAB function for minimizing nonlinear multivariable functions, and WORHP is a software libary for solving continuous large-scale NLPs. Applications for ICLOCS2 include robotics, automotive, aerospace, and communications [32].

In terms of using ICLOCS2, the user must first define the optimal control problem in a problem definition file, with elements such as final time, unknown constant parameters, initial conditions and guesses, state variables, and the number of control actions. Overall, there are 20 inputs required to define the problem and the cost function. For users who are not well-versed in MATLAB arrays nor optimal control problem set-up, the input of such parameters can become tedious.

Next, the user selects the solver settings in a separate file. This includes the transcription method, the derivative generation option (analytical or numerical), the perturbation, the output settings, and the type of NLP solver.

When the algorithm is run, the data from the problem definition and settings is passed to two functions. The first function, *transcribeOCP*, processes the information by defining the bounds, generates the initial guesses, constructs the Jacobian, generates the perturbation sets, formats the matrices for direct transcription, and defines the equations of motion, path constraints, and boundary constraints. The second function, *solveNLP*, solves the optimization problem using a solver such as *fmincon* or IPOPT (as specified by the user) and outputs the solution [31].

GPOPS II

GPOPS-II is a popular general-purpose MATLAB software that utilizes *Legendre-Gauss-Radau* quadrature orthogonal collocation to transcribe optimal control problems into large sparse NLPs. After transcription, the optimal control problem can be solved using the included IPOPT NLP solver or by interfacing GPOPS-II with the SNOPT NLP solver. GPOPS-II features an adaptive mesh refinement method that improves accuracy by increasing the flexibility in number and placement of mesh intervals. It also computes all required derivatives from the optimal control functions using sparse finite-differencing [28, 36]. The software allows the user to include integral constraints and boundary conditions, and can be used ”out-of-the-box” with MATLAB. In [36], the authors utilize GPOPS-II to solve several optimal control problems from open literature, to include a reusable launch vehicle entry problem, a space station attitude problem, and a multiple-stage launch vehicle ascent problem.

Similar to ICLOCS2, the user must define the optimal control problem, which requires knowledge of how to program the problem parameters as MATLAB structures and arrays of structures [36]. For the *input* structure, the user defines the name of the problem, the name of the continuous or endpoint function, the bounds structure, and a structure containing the initial guess. The user also has the option to provide auxillary data, derivative approximations for the NLP solver, scales, collocation method, mesh refinement method, type of NLP solver, and display level in the form of structures. The user is responsible for creating each input structure. The *gpops* function is called to obtain the solution via pseudospectral methods and either the SNOPT or IPOPT solver. The output structure contains the solution, the objective, the setup structure, the setup structure that would have been used with the next mesh refinement iteration, the error estimate for each mesh, and the number of mesh refinement iterations [36].

DIDO

Developed by Elissar Global, DIDO is a MATLAB-based optimal control software package that became more prominent in 2007 when NASA utilized it to execute the Zero Propellant Maneuver (ZPM), a globally optimal maneuver used to change the orientation of the International Space Station by 180 degrees without the need for propellant [37, 54]. Since 2001, DIDO has evolved from the use of a simple nonlinear programming solver to a more complex algorithm that is extremely robust, exhibits spectral acceleration based on discrete cotangent tunneling, and has verifiable accuracy. DIDO is not only the first pseudospectral solver introduced to the market, but was also the first guess-free, general-purpose optimal control problem software [37]. It is based on pseudospectral methods and boasts of being the ”minimalist’s approach” to solving optimal control problems. This is because the user only needs to provide the problem formulation and does not need to have any prior knowledge of pseudospectral methods [28, 37]. It acts similarly to both direct and indirect methods, in the sense that it provides costates and multipliers obtained from Pontryagin’s Principle like an indirect method, but also does not require user-supplied necessary conditions for optimality like a direct method. DIDO uses three main algorithms to obtain a solution, a stabilization component, an acceleration component, and an accuracy component, as well as its Hamiltonian programming method, which treats the control variable differently than the state variable [37].

Other Available Software Tools

In addition to ICLOCS2, GPOPS II, and DIDO, there are a vast number of other optimal control software packages available on the market. However, to the best of my knowledge, none utilize the CRBF approach to solving optimal control problems. The following table summarizes some of the other common software tools as well as their features, collocation method, and underlying solvers

[28]:

Table 1.1: Optimal Control Software Tools

Software Tool Name	**Features**	**Collocation Method**	**Underlying Solver**
ACADO	Automatic control, dynamic optimization, open source, code extensibility	Direct shooting methods [38]	Sequential Quadratic Programming (SQP)
DynOpt	Dynamic, optimization, user-defined constraints	Total discretization via orthogonal collocation [39]	SQP
OptimTraj	Optimal trajectory solver, user-defined constraints	Hermite-Simpson Direct Collocation, Orthogonal Collocation, 4th order Runge-Kutta [40]	Unspecified NLP
POCP	Polynomial problem data-based optimal control solver	Occupation measures/semidefinite programming relaxations [41]	SeDuMi, YALMIP's solvers
RIOTS	Finite-time, free final time, and variable initial condition problem solver	Sequential quadratic programming, projected descent method, augmented Lagrangian method [42]	Discrete-time solver
TOMLAB/PROPT	Global optimization, many available solvers	Pseudospectral methods [43]	SQP, MINOS, SNOPT, many

CHAPTER 2: METHODOLOGY

Optimal Control Problem Formulation

A nonlinear dynamic system can be described by the following equation:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t), \qquad t_0 \leq t \leq t_f \tag{2.1}$$

where the initial time $\boldsymbol{t_0}$ is given, the state $\boldsymbol{x}(t) \in \boldsymbol{R^n}$ and the control input $\boldsymbol{u}(t) \in \boldsymbol{R^m}$.

Consider the case of a continuous-time optimal control problem with a fixed terminal time and no terminal constraints. The optimal control problem can be written in the Bolza form with the goal of finding the control action $\boldsymbol{u}(t)$ that derives the states $\boldsymbol{x}(t)$ to minimize the scalar performance index:

$$J = \Phi(\boldsymbol{x}(t_f), t_f) + \int_{t_0}^{t_f} \mathscr{L}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) dt \tag{2.2}$$

subject to the dynamics and the path and controls constraints

$$\begin{aligned} \boldsymbol{P}(\boldsymbol{x}(t), t) &\leq \mathbf{0} \\ \boldsymbol{C}(\boldsymbol{u}(t), t) &\leq \mathbf{0} \end{aligned} \tag{2.3}$$

where, $\Phi(.)$ is the boundary terminal function and $\mathscr{L}(.)$ is the Lagrange term.
By defining the scalar Hamiltonian function as:

$$\mathscr{H} = \mathscr{L}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) + \boldsymbol{\lambda}^T \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \tag{2.4}$$

the augmented performance index can be rewritten as follows:

$$J = \Phi(\boldsymbol{x}(t_f), t_f) + \int_{t_0}^{t_f} (\mathscr{H} - \boldsymbol{\lambda}^T \dot{\boldsymbol{x}}) dt \tag{2.5}$$

The necessary conditions for optimality, or Euler-Langrange equations, can then be derived from the augmented performance index by applying the calculus of variations. [3, 22]

$$\begin{aligned} \dot{\boldsymbol{x}} &= \frac{\partial \mathscr{H}}{\partial \boldsymbol{\lambda}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t) \\ -\dot{\boldsymbol{\lambda}} &= \frac{\partial \mathscr{H}}{\partial \boldsymbol{x}} = \frac{\partial \mathscr{L}}{\partial \boldsymbol{x}} + \left[\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right]^T \boldsymbol{\lambda} \\ \boldsymbol{0} &= \frac{\partial \mathscr{H}}{\partial \boldsymbol{u}} = \frac{\partial \mathscr{L}}{\partial \boldsymbol{u}} + \left[\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right]^T \boldsymbol{\lambda} \end{aligned} \tag{2.6}$$

where $\boldsymbol{\lambda}$ is the Lagrange multipliers vector [45]. The initial and final states for the optimal control problem are then used to obtain the complementary boundary conditions [44].

In order to solve optimal control problems via collocation with CRBFs as the trial functions, it should first be expressed in the Bolza form as in Equation (2.2) with the defined dynamics, path and controls constraints expressed according to Equations (2.1) and (2.3). Next, the necessary conditions for optimality are obtained as in Equation (2.6). The resulting TPBVP is then transcribed into a system of nonlinear algebraic equations (NAEs) by using CRBFs, and the states, costates, control, and boundary conditions are approximated. A solver or numerical methods can then be used to solve the system of NAEs.

There are different types of RBFs and CRBFs that can be used as trial functions for collocation of optimal control problems. CRBFs are a new class of real-valued functions called Radial Basis Functions (RBFs) that are combined with a conical spline r^5. Along with a simple nodal distribution, this combination of RBFs with the conical spline provides the advantage of insensitivity to the shape parameter while maintaining accuracy and robustness. The shape parameter scales the flatness of an RBF by a value $c > 0$. The value of the RBF also depends on a set of approximation nodes t and the center node t_i, as well as the distance r between the approximation nodes. The table below provides the definitions for four classical RBFs, along with their associated CRBF when combined with a conical spline r^5.

Table 2.1: RBFs and CRBFs Definitions

Type	$\varphi(r,c)$ RBFs	$\varphi(r,c)$CRBFs
Multiquadric	$\sqrt{1+(cr)^2}$	$F + r^5$
Inverse multiquadric	$1/\sqrt{1+(cr)^2}$	$1/F + r^5$
Gaussian	$e^{-(cr^2)}$	$e^{-h^2} + r^5$
Inverse Quadratic	$1/(1+(cr)^2)$	$1/F^2 + r^5$

where $r = ||\boldsymbol{t}(i) - t_i||$, $h = \frac{r}{c}$ and $F = \sqrt{h^2+1}$.

After the appropriate CRBF is selected and the optimal control problem is expressed in the forms of Equations (2.2), (2.3), and (2.1), the states, control and costates can be approximated as:

$$\begin{aligned} \boldsymbol{x}(t) &\simeq \bar{\boldsymbol{x}}(r) = \sum_{i=1}^{N} \boldsymbol{\alpha}_i \varphi_i(r) \\ \boldsymbol{u}(t) &\simeq \bar{\boldsymbol{u}}(r) = \sum_{i=1}^{N} \boldsymbol{\beta}_i \varphi_i(r) \\ \boldsymbol{\lambda}(t) &\simeq \bar{\boldsymbol{\lambda}}(r) = \sum_{i=1}^{N} \boldsymbol{\gamma}_i \varphi_i(r) \end{aligned} \tag{2.7}$$

where, $i = 1, 2, \ldots, N$, $\bar{(.)}$ denotes an approximated function using CRBFs. The unknown coefficients are captured by vectors $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$.

The time-derivatives of the approximated states and costates, equation (2.7), are defined as:

$$\begin{aligned}\dot{\boldsymbol{x}}(r) &= \sum_{i=1}^{N} \boldsymbol{\alpha}_i \dot{\varphi}_i(r) \\ \dot{\boldsymbol{\lambda}}(r) &= \sum_{i=1}^{N} \boldsymbol{\gamma}_i \dot{\varphi}_i(r)\end{aligned} \tag{2.8}$$

For a univariate function with a single independent variable, in this case time, the first time-derivative of the CRBF function is

$$\frac{\partial \varphi}{\partial t} = \begin{cases} \frac{\partial \varphi(r)}{\partial r}\frac{\partial r}{\partial t}, & r \neq 0 \\ 0, & r = 0 \end{cases} \tag{2.9}$$

and the second time-derivative of φ is

$$\frac{\partial^2 \varphi}{\partial t^2} = \begin{cases} \frac{\partial^2 \varphi(r)}{\partial r^2}\left(\frac{\partial r}{\partial t}\right)^2 + \frac{\partial \varphi(r)}{\partial r}\frac{\partial^2 r}{\partial t^2}, & r \neq 0 \\ \frac{1}{3}c^2, & r = 0 \end{cases} \tag{2.10}$$

Next, an approximation matrix $\boldsymbol{\varphi}$ is defined based on the CRBF selected from 2.1 and the number of collocation points N. Each element of the approximation matrix $\boldsymbol{\varphi}$ is computed and the final matrix is expressed as:

$$\boldsymbol{\varphi} = \begin{bmatrix} \varphi(||t_1 - t_1||) & \varphi(||t_2 - t_1||) & \varphi(||t_3 - t_1||) & \dots & \varphi(||t_N - t_1||) \\ \varphi(||t_1 - t_2||) & \varphi(||t_2 - t_2||) & \varphi(||t_3 - t_2||) & \dots & \varphi(||t_N - t_2||) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \varphi(||t_1 - t_N||) & \varphi(||t_2 - t_N||) & \varphi(||t_3 - t_N||) & \dots & \varphi(||t_N - t_N||) \end{bmatrix} \tag{2.11}$$

After approximating the states, costates, and control, the necessary conditions in a compact form can be approximated. Recall equation (2.7)

$$\boldsymbol{\alpha} = \boldsymbol{\varphi}^{-1}\bar{\boldsymbol{x}} \tag{2.12}$$

taking the time-derivative

$$\dot{\bar{\boldsymbol{x}}} = \dot{\boldsymbol{\varphi}}\boldsymbol{\alpha} \tag{2.13}$$

Therefore

$$\dot{\bar{\boldsymbol{x}}} = \boldsymbol{D}\bar{\boldsymbol{x}} \tag{2.14}$$

where, $\boldsymbol{D} = \dot{\boldsymbol{\varphi}}\boldsymbol{\varphi}^{-1}$. Similarly, the costates time-derivatives are expressed as $\dot{\bar{\boldsymbol{\lambda}}} = \boldsymbol{D}\bar{\boldsymbol{\lambda}}$. Substituting equations (2.9) and (2.14) in equation (2.6) leads to an approximate form of the necessary conditions of optimality

$$\begin{aligned} \dot{x} \simeq \dot{\bar{x}} &= \boldsymbol{D}\bar{x} = f(\bar{x}, \bar{u}, r) \\ -\dot{\lambda} \simeq -\dot{\bar{\lambda}} &= \boldsymbol{D}\bar{\lambda} = \frac{\partial \mathscr{L}(\bar{x}, \bar{u}, r)}{\partial \bar{x}} + \frac{\partial f(\bar{x}, \bar{u}, r)}{\partial \bar{x}} \\ 0 &\simeq \frac{\partial \mathscr{L}(\bar{x}, \bar{u}, r)}{\partial \bar{u}} + \frac{\partial f(\bar{x}, \bar{u}, r)}{\partial \bar{u}} \end{aligned} \tag{2.15}$$

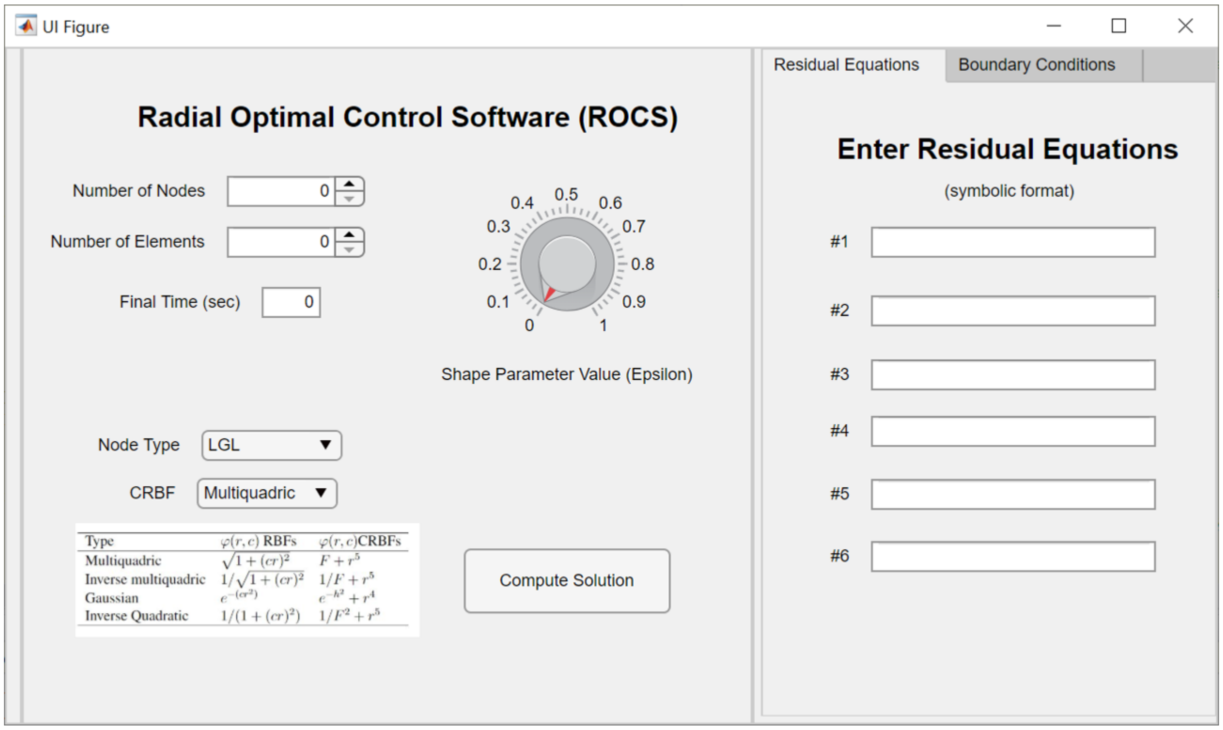
By utilizing a standard solver for nonlinear equations, the system of nonlinear equations (2.15) can be solved directly for the states, costates and controls.

Unified MATLAB Toolbox and Algorithm

For this research, a MATLAB software toolbox named Radial Optimal Control Software (ROCS) is introduced. ROCS not only implements the CRBF method, but also enables any user, from professionals to students, to solve nonlinear optimal control problems through a user-friendly interface. The tool accepts user input for boundary conditions, necessary conditions, and the governing equations of motion. In addition, the user can choose the number of collocation nodes and segments to satisfy the error tolerance defined by the user. The main feature of ROCS is utilizing adaptive local collocation methods using CRBFs to provide accurate and fast solutions to nonlinear optimal control problems.

ROCS's method for solving optimal control problems is based on that described in the previous section. First, the optimal control problem is expressed in the Bolza form as in Equation (2.2) with the defined dynamics, path and controls constraints according to Equations (2.1) and (2.3). Next, the necessary conditions for optimality are obtained, as in Equation (2.6), and the user specifies the boundary condition values. CRBF approximation is used to obtain a system of NAEs that are written in the form of a residual vector. The residual vector, necessary conditions, and boundary conditions are inputted into ROCS. From here, the user can specify the number of nodes, elements, CRBF type, and the nodal distribution. When the toolbox is run after inputting the initial parameters, ROCS evaluates the residual vector, derives the analytical Jacobian, and solves the system of NAEs using the built-in MATLAB solver *fsolve*. One major benefit of ROCS is the user can tailor the initial parameters, such as the number of nodes, the final time, and the boundary conditions, to obtain the best solution to the optimal control problem at hand. This gives the user the power to quickly change and update problem parameters without extensive manipulation or knowledge of code.

The user interface for ROCS is shown in Figure 2.1.



Radial Optimal Control Software (ROCS)

Number of Nodes: 0

Number of Elements: 0

Final Time (sec): 0

Shape Parameter Value (Epsilon)

Node Type: LGL

CRBF: Multiquadric

Enter Residual Equations

(symbolic format)

Type	$\varphi(r,c)$ RBFs	$\varphi(r,c)$CRBFs
Multiquadric	$\sqrt{1+(cr)^2}$	$F+r^5$
Inverse multiquadric	$1/\sqrt{1+(cr)^2}$	$1/F+r^5$
Gaussian	$e^{-(cr^2)}$	$e^{-h^2}+r^4$
Inverse Quadratic	$1/(1+(cr)^2)$	$1/F^2+r^5$

Figure 2.1: ROCS Interface

The ROCS toolbox was built using MATLAB Apps, which allows for graphical user interface (GUI) design with underlying code to perform algorithms, collect data, and output results. Through MATLAB Apps, the ROCS toolbox is packaged with all the files and code necessary to operate it. This allows it to be shared easily with other users through a public repository or any other file exchange website.

The toolbox is made available to MATLAB users via GitHub, a website for distribution of open source code. GitHub is a popular online software development and code sharing platform that allows developers to to share projects to an open source community. From the GitHub website, users can download the ROCS toolbox and associated documentation. Instructions for downloading ROCS through GitHub are included in APPENDIX: INSTRUCTIONS FOR DOWNLOADING

ROCS TOOLBOX.

In the next section, the ROCS toolbox's CRBF-collocation method is illustrated in detail by applying it to three nonlinear optimal control problems.

CHAPTER 3: FINDINGS

To demonstrate the usefulness and application of ROCS, three nonlinear optimal control problems are solved using the CRBF-collocation method and the results are compared to the solution found in literature. For this research, the three problems examined are Lambert's problem (two-body orbital transfer), Zermelo's problem, and the fixed state duffing oscillator problem.

Lambert's Problem (Two-Body Orbital Transfer)

The classical Lambert's problem, or two-body orbital transfer problem, arose as a celestial mechanics problem in the 18th century. Named after Johann Heinrich Lambert (1728-1777) who made significant contributions to the problem's solution, Lambert's problem objective is to determine the Keplerian orbit associated with two positions in space at a given time t. This problem is a popular celestial mechanics topic due to its applications in trajectory design, navigation, interplanetary transfer design, and missile and spacecraft targeting [46]. According to Lambert, "The transfer time Δt of a body moving between two points r_1 and r_2 on a conic trajectory is a function only of the sum r_1 + r_2 of the distances of the two points from the origin of the force, the linear distance c between the points, and the semi-major axis a of the conic section" [46]. The geometry for Lambert's problem is shown in Figure 3.1.

For this work, the unperturbed relative two-body problem is solved. The dynamics can be obtained from Newton's universal gravitational law, where the position vector is $\mathbf{r} = [x \; y \; z]^T$ and $\mu \approx 3.986 \cdot 10^{14} \; m^3 s^{-2}$:

$$\ddot{\mathbf{r}} = \frac{-\mu}{\mathbf{r^3}}\mathbf{r} \tag{3.1}$$

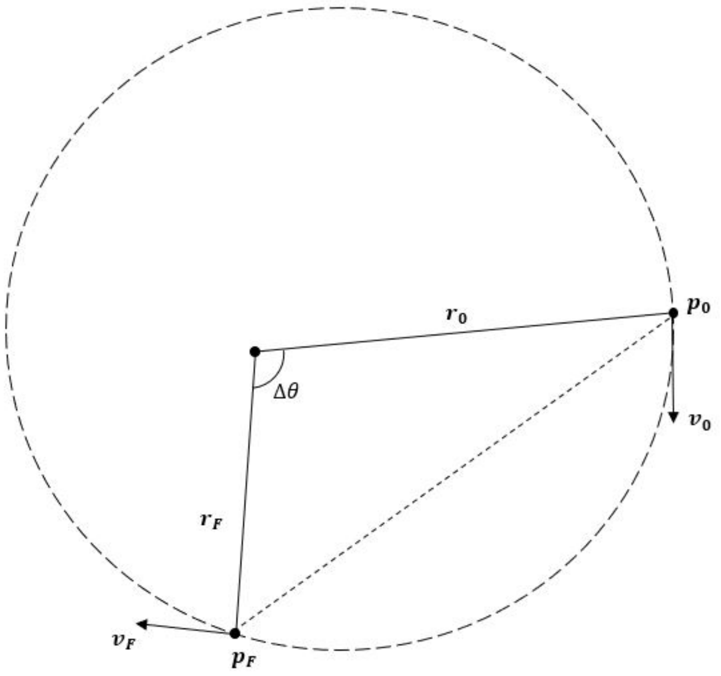


Figure 3.1: Lambert's Problem Geometry

Lambert's problem can be solved analytically [47] or through numerical iterative methods [48].

The problem can be cast into a system of first-order differential equations:

$$\begin{aligned} \dot{x_1} &= x_2 \\ \dot{x_2} &= \frac{-\mu}{r^3} x_1 \\ \dot{y_1} &= y_2 \\ \dot{y_2} &= \frac{-\mu}{r^3} y_1 \\ \dot{z_1} &= z_2 \\ \dot{z_2} &= \frac{-\mu}{r^3} z_1 \end{aligned} \tag{3.2}$$

Furthermore, the system of NAEs for Lambert's problem can be rewritten as a system of 6*N* equations in a residual form [22]:

$$\begin{aligned}
\boldsymbol{R}_1^i &= \boldsymbol{D}x_1^i - x_2^i \\
\boldsymbol{R}_2^i &= \boldsymbol{D}y_1^i - y_2^i \\
\boldsymbol{R}_3^1 &= \boldsymbol{D}z_1^i - z_2^i \\
\boldsymbol{R}_4^1 &= x_1^1 - x_{10} \\
\boldsymbol{R}_4^j &= \boldsymbol{D}x_2^j + \frac{\mu}{r^j}x_1^j \\
\boldsymbol{R}_4^N &= x_1^N - x_{1F} \\
\boldsymbol{R}_5^1 &= y_1^1 - y_{10} \\
\boldsymbol{R}_5^j &= \boldsymbol{D}y_2^j + \frac{\mu}{r^j}y_1^j \\
\boldsymbol{R}_5^N &= y_1^N - y_{1F} \\
\boldsymbol{R}_6^1 &= z_1^1 - z_{10} \\
\boldsymbol{R}_6^j &= \boldsymbol{D}z_2^j + \frac{\mu}{r^j}z_1^j \\
\boldsymbol{R}_6^N &= z_1^N - z_{1F}
\end{aligned} \tag{3.3}$$

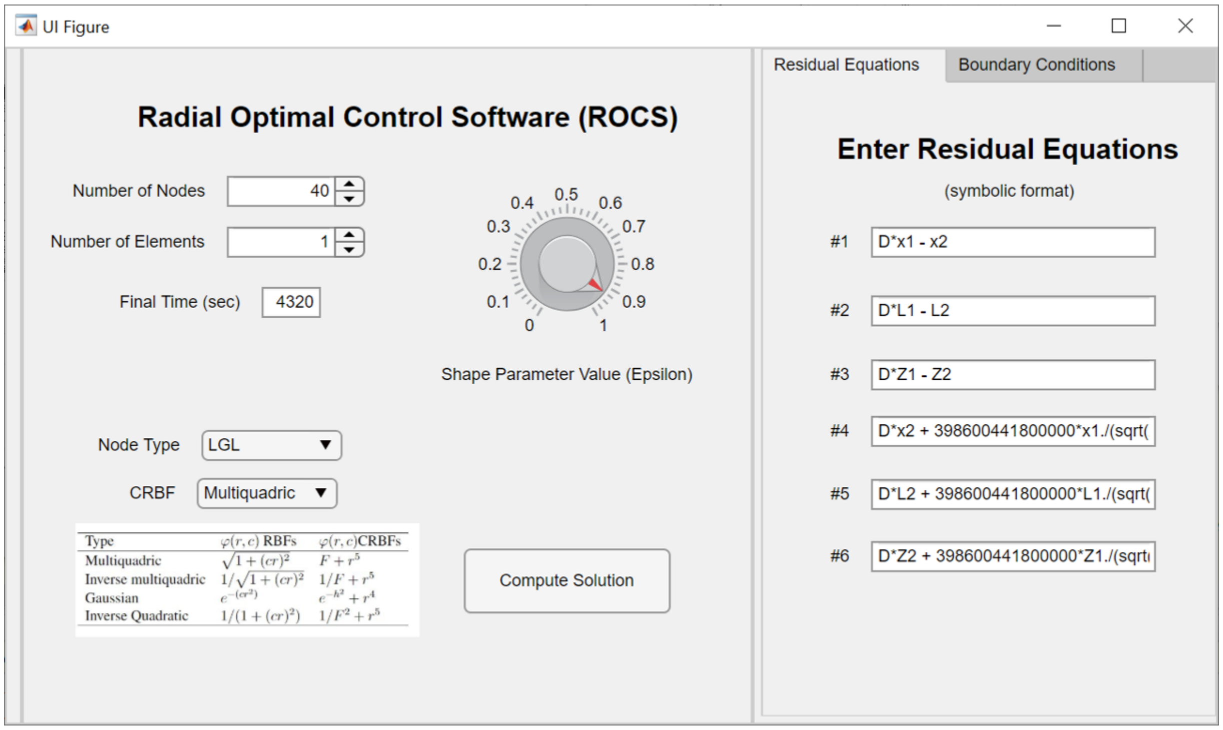
where, *i* = 1, ..., *N* and *j* = 2, ..., *N*-1.

To solve the two-body orbital transfer problem, the initial parameters displayed in Table 3.1 were used. The initial conditions selected for this problem are $r_0 = [2.87\ 5.19\ 2.85]^T$ x 10^6 m and the final conditions are $r_f = [2.09\ 7.82\ 0]^T$ x 10^6 m. In terms of nodes, the number chosen was 40 and a multiquadric CRBF $\varphi(r, c)$ is selected, where $r = ||\boldsymbol{t}(i) - t_i||$, $F = \sqrt{h^2 + 1}$, and $h = \frac{r}{c}$. The transfer time, or final time, was selected to be t_f = 4.32 x 10^3 seconds, or t_f = 0.05 days.

Table 3.1: Lambert's Problem Initial Parameters

Initial Conditions	Value
N (nodes)	40
ϵ (shape parameter)	0.9
r_0	$[2.87\ \ 5.19\ \ 2.85]^T$ x 10^6 m
r_f	$[2.09\ \ 7.82\ \ 0]^T$ x 10^6 m
t_f	4.32 x 10^3 sec
$\varphi(r, c)$ (CRBF)	Multiquadric $(F + r^5)$

Once the optimal conditions have been derived and Lambert's problem has been rewritten in residual form, as shown in Equation 3.3, they are inputted into ROCS along with the boundary conditions. The user interface for ROCS after inputting the problem parameters is shown in Figure 3.2.



Radial Optimal Control Software (ROCS)

Number of Nodes 40

Number of Elements 1

Final Time (sec) 4320

Shape Parameter Value (Epsilon)

Node Type LGL

CRBF Multiquadric

Type	$\varphi(r,c)$ RBFs	$\varphi(r,c)$CRBFs
Multiquadric	$\sqrt{1+(cr)^2}$	$F+r^5$
Inverse multiquadric	$1/\sqrt{1+(cr)^2}$	$1/F+r^5$
Gaussian	$e^{-(cr^2)}$	$e^{-h^2}+r^4$
Inverse Quadratic	$1/(1+(cr)^2)$	$1/F^2+r^5$

Enter Residual Equations

(symbolic format)

#1	D*x1 - x2
#2	D*L1 - L2
#3	D*Z1 - Z2
#4	D*x2 + 398600441800000*x1./(sqrt(
#5	D*L2 + 398600441800000*L1./(sqrt(
#6	D*Z2 + 398600441800000*Z1./(sqrt

Figure 3.2: ROCS Interface for Lambert's Problem

ROCS solves the set of NAEs shown in Equation 3.3 using the CRBF-collocation method. To determine the accuracy of ROCS's solution, the closed form of the Lagrange/Gibbs (FG) solution to Lambert's problem is obtained using the same initial parameters. The comparison between the CRBF-collocation method and the closed-form solution is shown in Figure 3.3 for x_1 and y_1, and Figure 3.4 for x_2 and y_2.

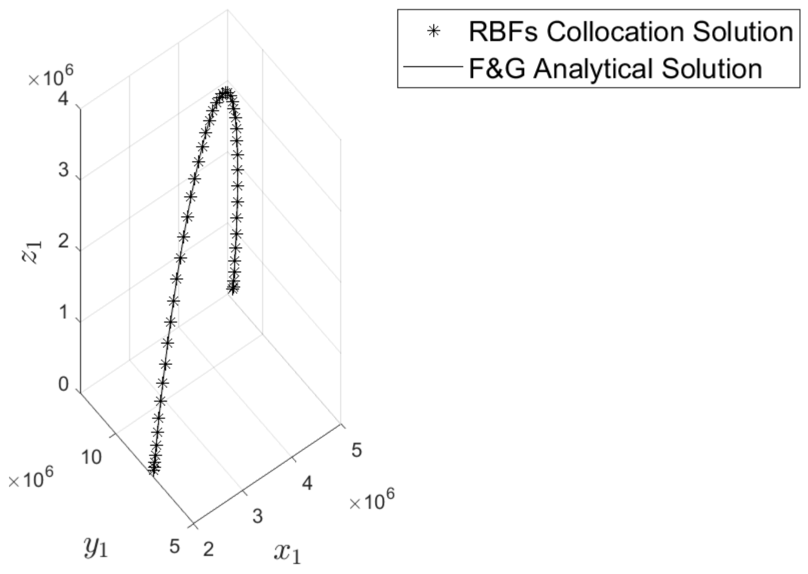


Figure 3.3: Lambert's Problem Solution for x_1 and y_1 : CRBF vs Closed-Form Solution

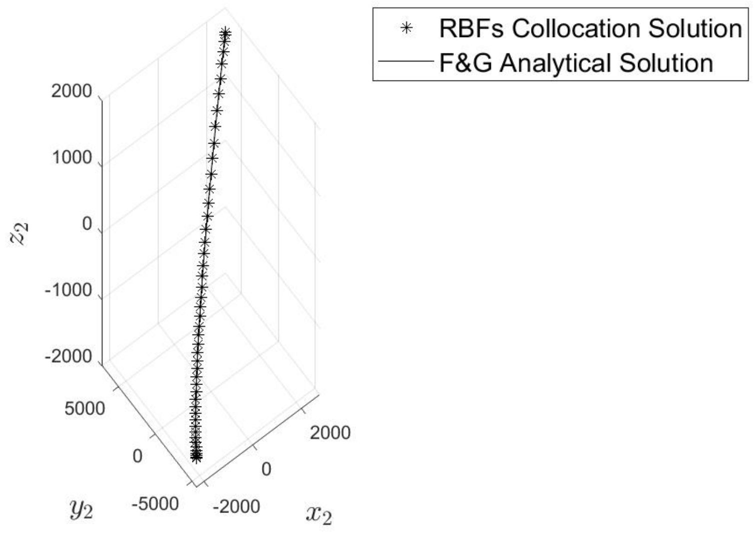


Figure 3.4: Lambert's Problem Solution for x_2 and y_2 : CRBF vs Closed-Form Solution

Zermelo's Problem

First proposed by Ernst Zermelo in 1931, Zermelo's problem is a classical optimal control problem where the goal is to find the minimum time for a vessel to navigate between two points in a flow [49, 50]. In this case, we consider a vessel navigating between an initial and final point through water with a linear distribution. The vessel shown in Figure 3.5 depicts Zermelo's problem, where V is the linearly applied wind velocity as the vessel travels from point A to point B at a constant velocity v.

In order to find the time and required bearing $\theta(t)$ it takes for the vessel to travel from point A to

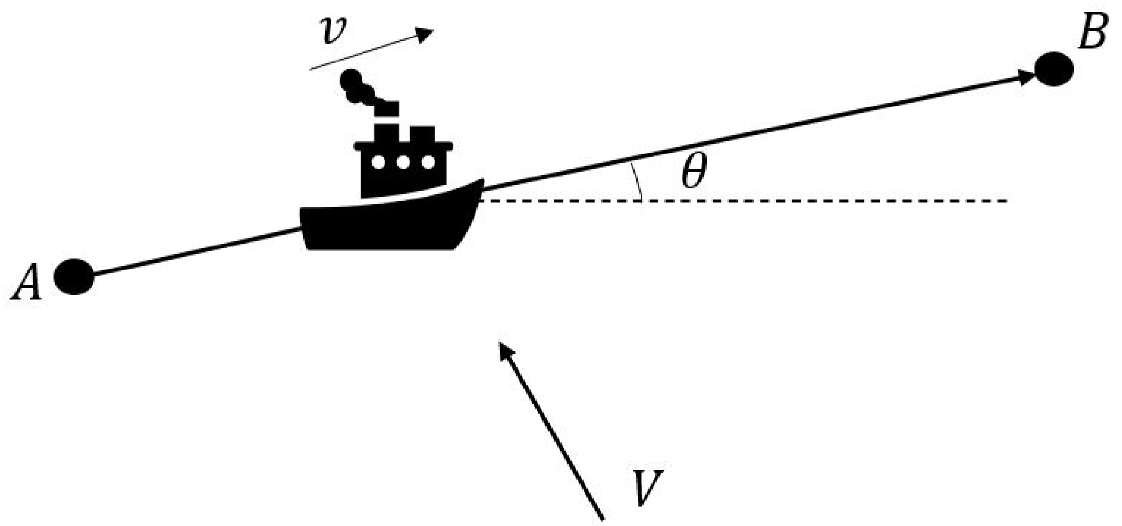


Figure 3.5: Zermelo's Navigation Problem

point B at a constant velocity v, we must minimize:

$$J = \int_{t_0}^{t_f} dt \tag{3.4}$$

subject to

$$\begin{aligned} \dot{x} &= V\cos\theta + \frac{Vy}{h} \\ \dot{y} &= V\sin\theta \end{aligned} \tag{3.5}$$

Equation 3.4 represents the total time it takes for the vessel to travel from its initial position to its final position. The total time is subject to (x, y) and $(\dot{x}, \dot{y})$, which are the vessel's position and velocity, respectively.

For comparison with solving the optimal control problem using CRBFs, there are two equations that provide the exact solution of Zermelo's problem. The required bearing and final bearing to minimize the total time can be determined based on the vessel's current position (x, y), as shown by the position equations (3.6). [45].

$$\begin{aligned} x(t) &= \frac{1}{2}\left(\sec\theta(t_f)(\tan\theta(t_f) - \tan\theta) - \tan\theta(\sec\theta(t_f) - \sec\theta)\right. \\ &\quad \left. + ln\frac{\sec\theta(t_f) + \tan\theta(t_f)}{\sec\theta + \tan\theta}\right) \\ \frac{y(t)}{h} &= \sec\theta(t_f) - \sec\theta(t) \end{aligned} \tag{3.6}$$

From the problem statement, the Hamiltonian can be written as:

$$\mathscr{H} = 1 + \lambda_x \left(V\cos\theta + \frac{Vy}{h} \right) + \lambda_y V \sin\theta \tag{3.7}$$

where the necessary conditions for optimality can be derived as:

$$\begin{aligned} -\dot{\lambda}_x &= \frac{\partial \mathscr{H}}{\partial x} = 0 \\ -\dot{\lambda}_y &= \frac{\partial \mathscr{H}}{\partial y} = \frac{\lambda_x V}{h} \\ 0 &= \frac{\partial \mathscr{H}}{\partial \theta} = -\lambda_x V \sin\theta + \lambda_y V \cos\theta \end{aligned} \tag{3.8}$$

From here, we can obtain the differential equations for Zermelo's problem:

$$\begin{aligned} \dot{x} &= V\cos\left(\frac{\lambda_y}{\lambda_x}\right) + \frac{Vy}{h}, \quad x_0 = 4.9, \quad x_f = 0 \\ \dot{y} &= V\sin\left(\frac{\lambda_y}{\lambda_x}\right), \quad y_0 = 1.6 \quad y_f = 0 \\ \dot{\lambda}_x &= 0 \\ \dot{\lambda}_y &= -\frac{V}{h}\lambda_x \end{aligned} \tag{3.9}$$

To approximate the differential equations (3.9) for Zermelo's problem, Equations (2.7), (2.9) and (2.10) are used. After approximating, the set of differential equations (3.9) can be rewritten in a residual vector form $\boldsymbol{R}$:

$$\begin{aligned}
\boldsymbol{R}_1^1 &= x^1 - x_0 \\
\boldsymbol{R}_1^i &= \boldsymbol{D}x^i - V\cos\left(\frac{\lambda_y^i}{\lambda_x}\right) - \frac{V}{h}y^i \\
\boldsymbol{R}_2^1 &= y^1 - y_0 \\
\boldsymbol{R}_2^i &= \boldsymbol{D}y^i - V\sin\left(\frac{\lambda_y^i}{\lambda_x}\right) \\
\boldsymbol{R}_3^j &= \boldsymbol{D}\lambda_y^j + \frac{V}{h}\lambda_x \\
\boldsymbol{R}_3^N &= y^N - y_f \\
\boldsymbol{R}_4 &= x^N - x_f \\
\boldsymbol{R}_5 &= 1 + V\lambda_x\cos\left(\frac{\lambda_y^N}{\lambda_x}\right) + \frac{V}{h}\lambda_x y^N + V\lambda_y^N\sin\left(\frac{\lambda_y^N}{\lambda_x}\right)
\end{aligned} \tag{3.10}$$

where $i = 2, \ldots, N$ and $j = 1, \ldots, N-1$. Since λ_x and t_f are constants, the residual vector $\boldsymbol{R}$ is of length $3N + 2$.

From the costates represented in Equation 3.8, the control input $\theta(t)$ is:

$$\theta = \tan^{-1}\left(\frac{\lambda_y}{\lambda_x}\right) \tag{3.11}$$

To solve Zermelo's problem using the CRBF-collocation implemented by the ROCS toolbox, a set of initial parameters are selected. Table 3.2 displays the initial parameters used.

Table 3.2: Zermelo's Problem Initial Parameters

Initial Conditions	Value
N (nodes)	25
ϵ (shape parameter)	1
t_f	5.693 sec
initial guess	7
initial position	(4.9, 1.66)
final position	(0,0)
$\varphi(r,c)$ (CRBF)	Multiquadric $(F + r^5)$

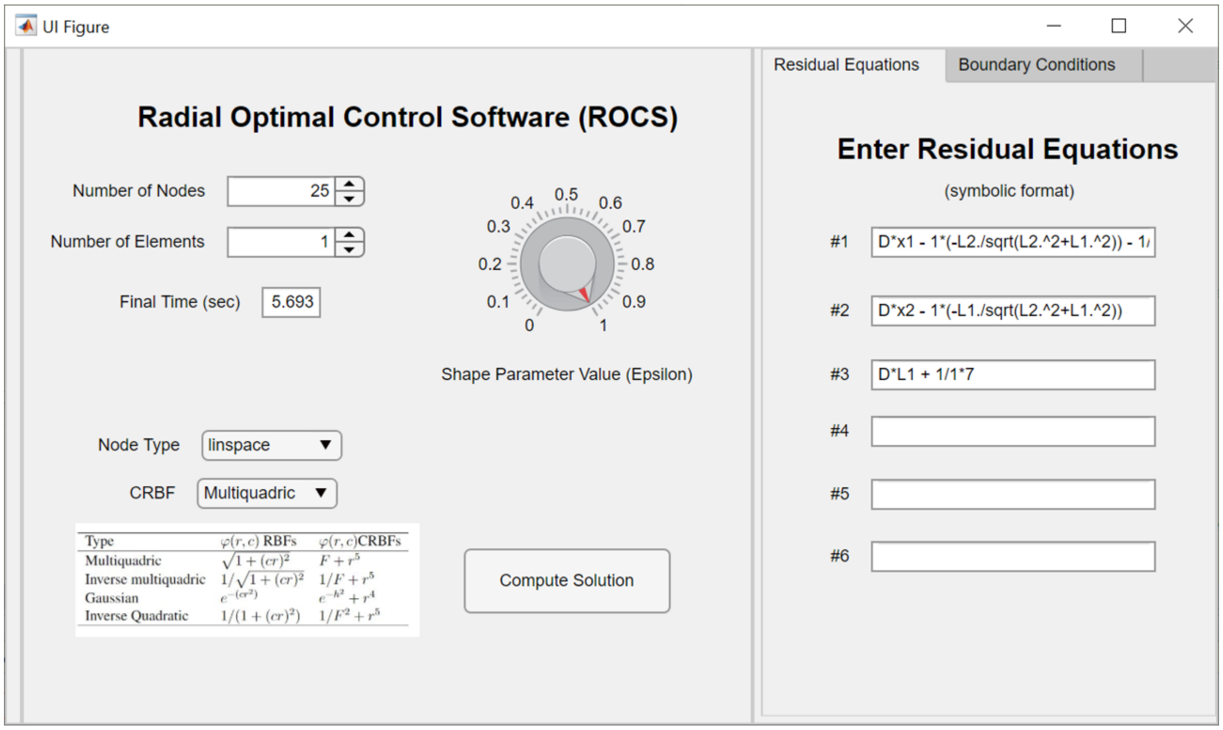
As displayed in Table 3.2, 25 equidistant nodes are used and an initial guess of 7 is selected for all the unknowns. The initial position is (4.9, 1.66) and the goal is to find the optimal trajectory to the position (0,0) in t_f = 5.693 seconds. A multiquadric CRBF $\varphi(r,c)$ from Table 2.1 is used for this problem, where $r = ||\boldsymbol{t}(i) - t_i||$, $F = \sqrt{h^2+1}$, and $h = \frac{r}{c}$.

The boundary conditions are collocated in the residual vector at the following residual vector indices:

$$\begin{aligned}\boldsymbol{R}_1^1 &= x^1 - x_0\\ \boldsymbol{R}_2^1 &= y^1 - y_0\\ \boldsymbol{R}_3^N &= x^N - x_f\\ \boldsymbol{R}_4^N &= y^N - y_f\end{aligned} \tag{3.12}$$

Collocating at these residual vector indices ensures the boundary conditions are enforced. In addition, since the Hamiltonian is a constant, the residual vector $\boldsymbol{R}_5$ includes the Hamiltonian function. By including the Hamiltonian in the vector of unknowns, ROCS can solve the problem with the final time being unknown.

After putting the Zermelo's problem in residual form and deriving the optimal conditions, they are



Radial Optimal Control Software (ROCS)

Number of Nodes: 25

Number of Elements: 1

Final Time (sec): 5.693

Shape Parameter Value (Epsilon)

Node Type: linspace

CRBF: Multiquadric

Type	$\varphi(r,c)$ RBFs	$\varphi(r,c)$CRBFs
Multiquadric	$\sqrt{1+(cr)^2}$	$F+r^5$
Inverse multiquadric	$1/\sqrt{1+(cr)^2}$	$1/F+r^5$
Gaussian	$e^{-(cr^2)}$	$e^{-h^2}+r^4$
Inverse Quadratic	$1/(1+(cr)^2)$	$1/F^2+r^5$

Enter Residual Equations

(symbolic format)

#1 D*x1 - 1*(-L2./sqrt(L2.^2+L1.^2)) - 1/

#2 D*x2 - 1*(-L1./sqrt(L2.^2+L1.^2))

#3 D*L1 + 1/1*7

#4

#5

#6

Figure 3.6: ROCS Interface for Zermelo's Problem

inputted into ROCS along with the boundary conditions as shown in Figure 3.6.

The set of residual vectors shown in Equations (3.10) are solved using *fsolve* within the ROCS code. Similarly, the *fsolve* function is used along with the built-in ode45 solver to obtain the exact solution from Equation (3.6). The relative and absolute tolerances of 10^{-20} are used. The comparison between the exact solution and the solution obtained via the CRBF-collocation method are shown in Figure 3.7.

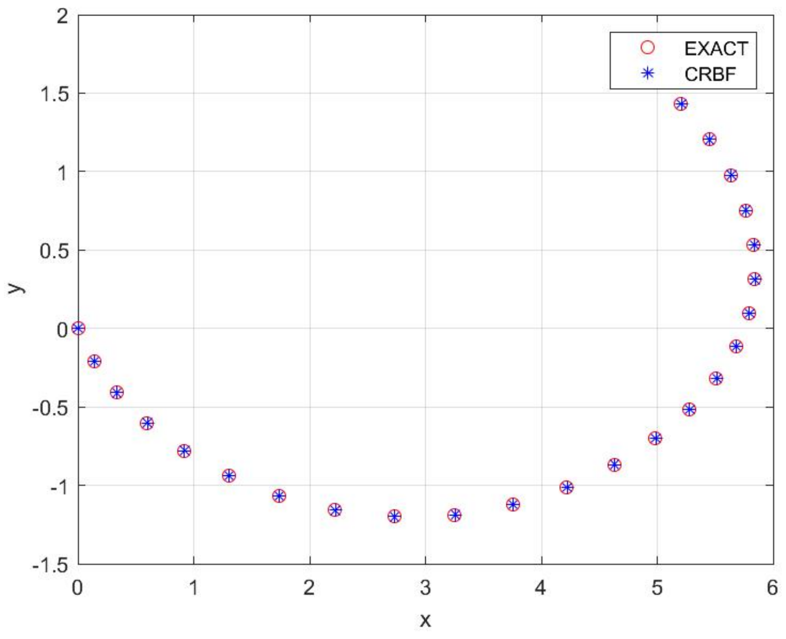


Figure 3.7: Zermelo's Problem Solution: CRBF vs Exact Solution

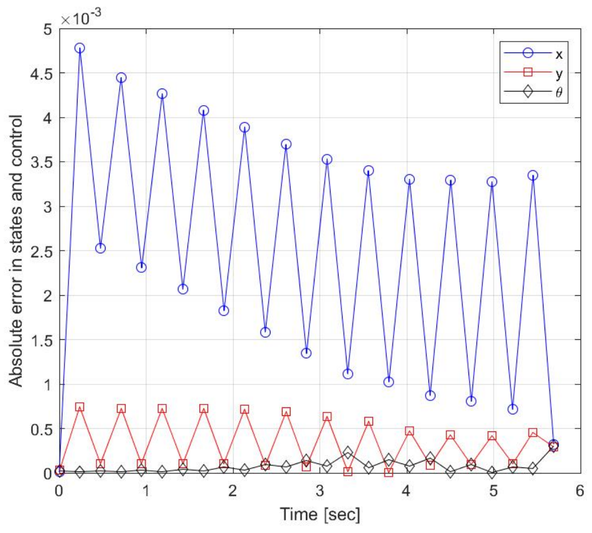


Figure 3.8: Error between ROCS Solution and Exact Solution

The absolute error between the states and control are shown in Figure 3.8. From the figure, the error is on the magnitude of 10^{-3}, which demonstrates the accuracy of ROCS when compared to the exact solution.

To demonstrate the CRBF-collocation method's insensitivity to the shape parameter, Zermelo's problem was solved with the same initial parameters as before, but with a shape parameters over a range of values: c = [1 10 100 1000]. The RMSE for the states and control as well as the heading angle are on the order of 10^{-4} shown in Figure 3.9.

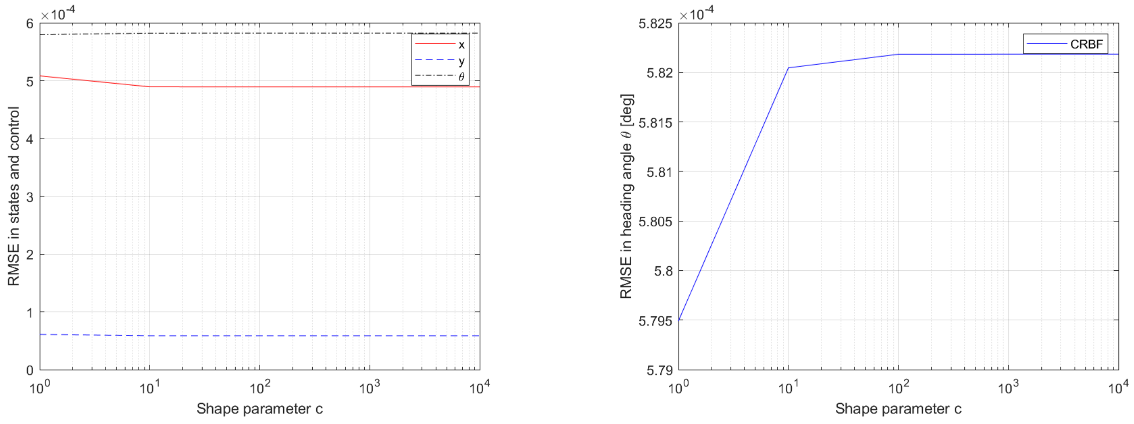


Figure 3.9: RMSE for States and Control, and Heading Angle

Duffing Oscillator

The duffing oscillator is a popular problem in engineering, mathematics, and physics because it can be used to approximate the behavior of different physical systems. Named after Georg Duffing, who published a book on the topic in 1918, the duffing equation is a second-order differential equation with a cubic nonlinearity [51]. The equation became more popular in the 1970s when researchers began using it to model chaotic system behavior [52]. The dynamics for the duffing oscillator are shown in Equation 3.13.

$$\ddot{x} + a^2 x + b x^3 = u, \qquad 0 \leq t \leq t_f \tag{3.13}$$

where a is the natural frequency, b is the nonlinearity coefficient, u is the unknown forcing function.

For the duffing oscillator problem, the cost functional to minimize is:

$$J = \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_f)\boldsymbol{S}(\boldsymbol{x} - \boldsymbol{x}_f) + \frac{1}{2}\int_{t_0}^{t_f} \boldsymbol{u}^2 dt \tag{3.14}$$

where $\boldsymbol{S} = \begin{bmatrix} s_{11} & s_{12} \\ s_{21} & s_{22} \end{bmatrix}$ is the terminal function weight matrix.

The system dynamics described by the duffing equation 3.13 can be rewritten as a system of first-order ordinary differential equations (ODEs):

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -a^2 x_1 - b x_1^3 + u \end{aligned} \tag{3.15}$$

where x_1 and x_2 denote the position and velocity, respectively.

To derive the nonlinear algebraic equations required to solve the problem using CRBFs, we first write the Hamiltonian as:

$$\mathscr{H} = \frac{1}{2}u^2 + \lambda_1 x_2 + \lambda_2(-a^2 x_1 - b x_1^3 + u) \tag{3.16}$$

Next, the necessary conditions for optimality can be written as:

$$\begin{aligned} \dot{\lambda}_1 &= \lambda_2(a^2 + 3bx_1^2) \\ \dot{\lambda}_2 &= -\lambda_1 \\ u &= -\lambda_2 \end{aligned} \tag{3.17}$$

The system of differential equations that describe the duffing oscillator can be found by combining equations (3.15) and (3.17) to reach a TPBVP:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -a^2x_1 - bx_1^3 - \lambda_2 \\ \dot{\lambda}_3 &= \lambda_2(a^2 + 3bx_1^2) \\ \dot{\lambda}_4 &= -\lambda_1 \end{aligned} \tag{3.18}$$

Using, Equations (2.7), (2.9) and (2.10), the set of differential equations (3.18) can be rewritten as nonlinear algebraic equations in a residual vector form $\boldsymbol{R}$:

$$\begin{aligned} \boldsymbol{R}_1^k &= Dx_1^k - x_2^k \\ \boldsymbol{R}_2^k &= Dx_2^k - (a^2x_1^k + b(x_1^k)^3 + \lambda_2^k) \\ \boldsymbol{R}_3^k &= D\lambda_3^k - \lambda_2^k(a^2 + 3b(x_1^k)^2) \\ \boldsymbol{R}_4^k &= D\lambda_4^k + \lambda_1^k \end{aligned} \tag{3.19}$$

where, $k = 1 \ldots N$ and $\boldsymbol{R}$ is the residual vector to be minimized.

To demonstrate the capabilities of ROCS, the duffing oscillator problem is solved for the fixed final state case.

Fixed Final State

For the fixed final state duffing oscillator problem, the initial parameters used are shown in Table 3.3.

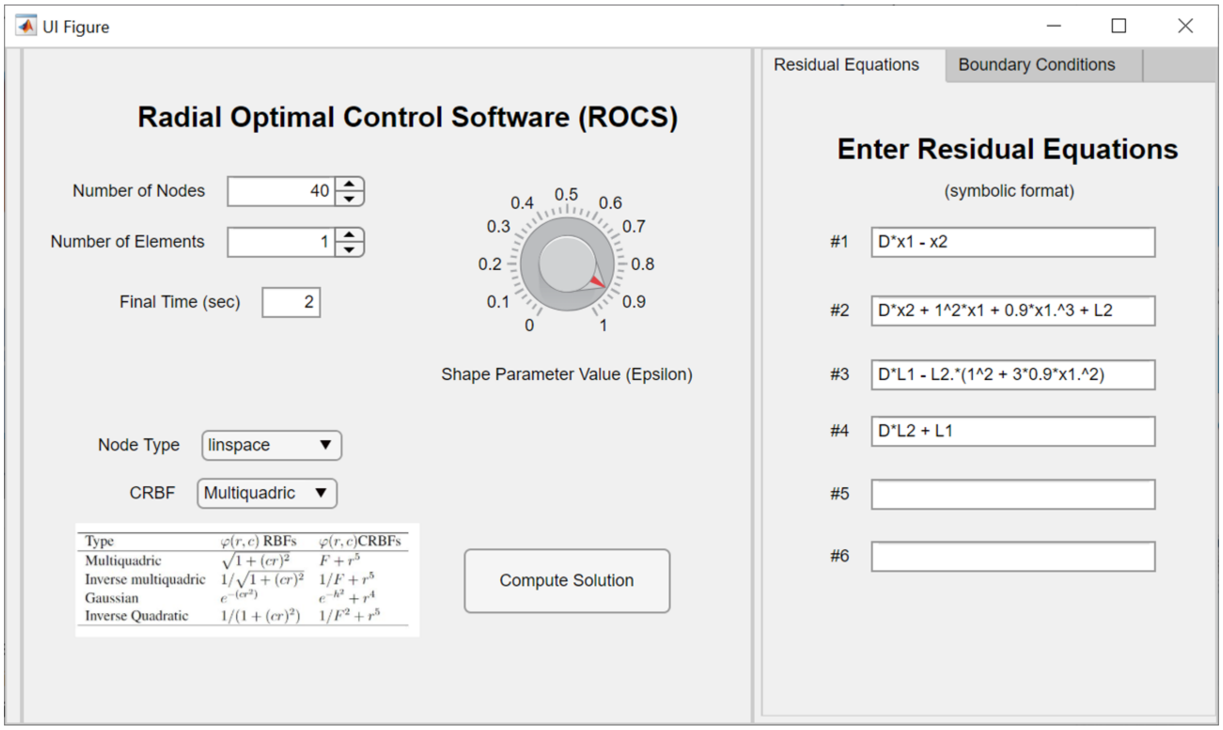
Table 3.3: Duffing Oscillator Problem Initial Parameters

Initial Conditions	Value
a (natural freq.)	1 rad/sec
b (nonlinearity coeff.)	0.9
N (nodes)	40
ϵ (shape parameter)	0.9
x_{1_0} (initial)	0
x_{2_0} (initial)	0
x_{1_f} (final)	2
x_{2_f} (final)	3
t_f	2 sec
$\varphi(r,c)$ (CRBF)	Multiquadric ($F + r^5$)

As displayed in Table 3.3, the natural frequency a is set to 1 rad/sec, while the nonlinearity coefficient b is set to 0.9. For this problem, 40 equidistant nodes are used and a multiquadric CRBF $\varphi(r,c)$ is selected, where $r = ||\boldsymbol{t}(i) - t_i||$, $F = \sqrt{h^2+1}$, and $h = \frac{r}{c}$. The initial conditions selected for this problem are $x_{1_0} = 0$ and $x_{2_0} = 0$ and the final conditions are x_{1_f} = 2 and x_{2_f} = 3.

To ensure that the boundary condition values are enforced, they are collocated at the following residual vector indices:

$$\begin{aligned}
\boldsymbol{R}_1^1 &= x_1^1 - x_{1_0} \\
\boldsymbol{R}_2^1 &= x_2^1 - x_{2_0} \\
\boldsymbol{R}_3^N &= x_1^N - x_{1f} \\
\boldsymbol{R}_4^N &= x_2^N - x_{2f}
\end{aligned} \tag{3.20}$$



Radial Optimal Control Software (ROCS)

Number of Nodes: 40

Number of Elements: 1

Final Time (sec): 2

Shape Parameter Value (Epsilon)

Node Type: linspace

CRBF: Multiquadric

Type	$\varphi(r,c)$ RBFs	$\varphi(r,c)$CRBFs
Multiquadric	$\sqrt{1+(cr)^2}$	$F+r^5$
Inverse multiquadric	$1/\sqrt{1+(cr)^2}$	$1/F+r^5$
Gaussian	$e^{-(cr^2)}$	$e^{-h^2}+r^4$
Inverse Quadratic	$1/(1+(cr)^2)$	$1/F^2+r^5$

Enter Residual Equations

(symbolic format)

```
#1  D*x1 - x2
#2  D*x2 + 1^2*x1 + 0.9*x1.^3 + L2
#3  D*L1 - L2.*(1^2 + 3*0.9*x1.^2)
#4  D*L2 + L1
#5
#6
```

Figure 3.10: ROCS Interface for Duffing Oscillator - Fixed Final State

After deriving the optimal conditions and formulating the duffing oscillator problem into a residual vector form, as shown by Equations 3.17 and 3.19, respectively, they are inputted into ROCS along with the boundary conditions as shown in Figure 3.10. It is important to note that when entering the residual equations, the natural frequency and nonlinearity coefficient are substituted accordingly. The final time is set to 2 seconds.

To demonstrate the accuracy of ROCS's calculations, the exact solution for the duffing oscillator is solved using the same conditions from the CRBF solution. The built-in MATLAB single-step solver *ode23* is utilized. The ode23 solver implements a Runge-Kutta (2,3) pair of Bogacki and Shampine [53]. The trajectories of the optimal states and control are plotted and shown in Figure 3.11. The absolute error between the states is 0.25 or less, whereas for the costates it remains below

1.4, as shown in Figure 3.12. Overall, the solution provided from the CRBF-collocation method implemented by ROCS displays little error compared to the exact solution generated by the *ode23* solver.

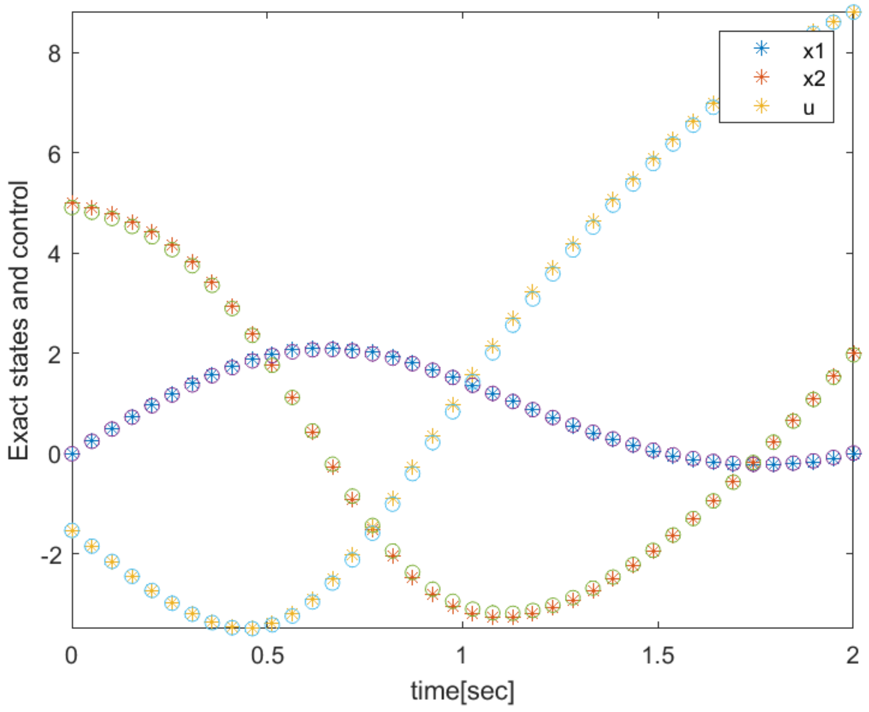


Figure 3.11: Fixed Final State Duffing Oscillator - Optimal States and Control with Multiquadric CRBF

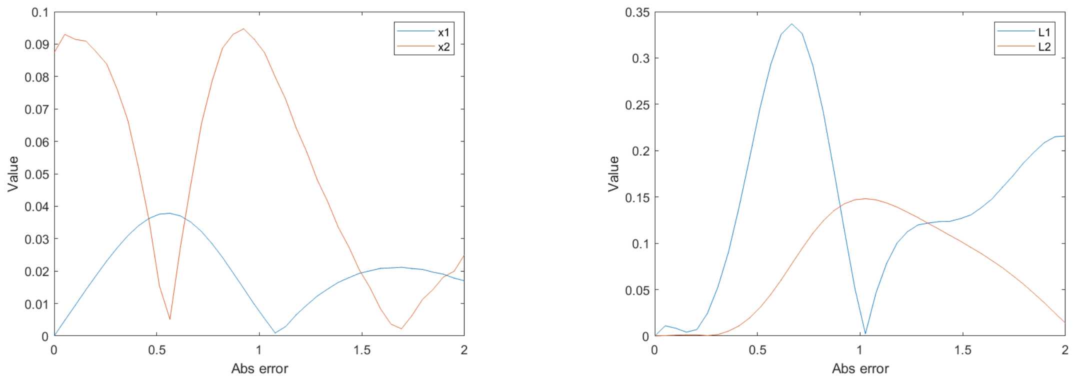


Figure 3.12: Absolute Error for States and Costates with Multiquadric CRBF

To demonstrate the performance of another CRBFs, the fixed final state duffing oscillator problem is solved using the Gaussian CRBF type, as shown in Table 2.1. The same initial parameters used with the Multiquadric CRBF solution are utilized for this solution. Figure 3.13 displays the trajectories of the optimal states and control, while Figure 3.14 displays the absolute error between the states and the costates when compared to the exact solution. Overall, like the Multiquadric CRBF, the Gaussian CRBF showed good accuracy compared to the exact solution generated by the ode23 solver.

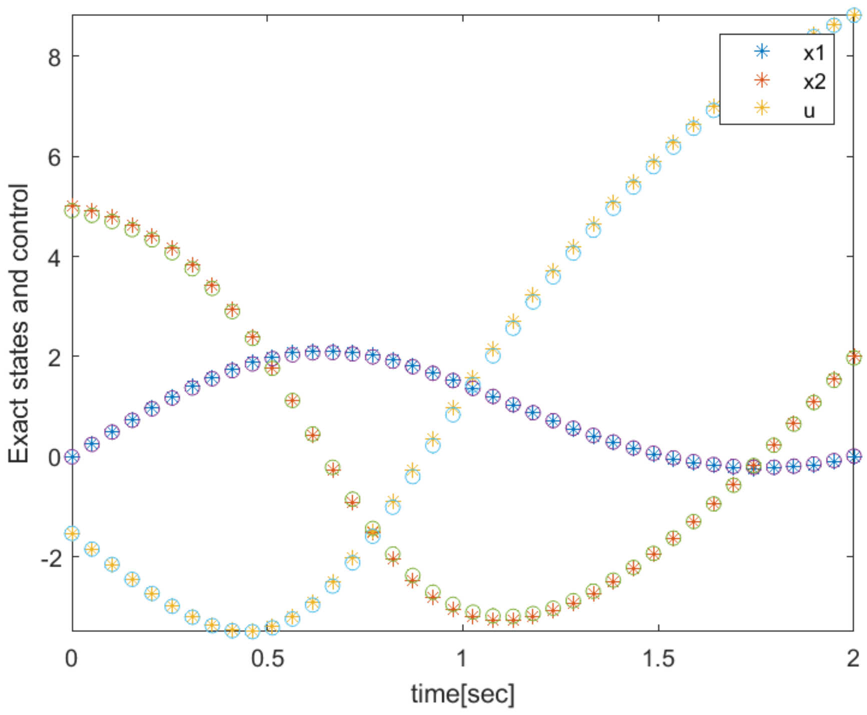


Figure 3.13: Fixed Final State Duffing Oscillator - Optimal States and Control with Gaussian CRBF

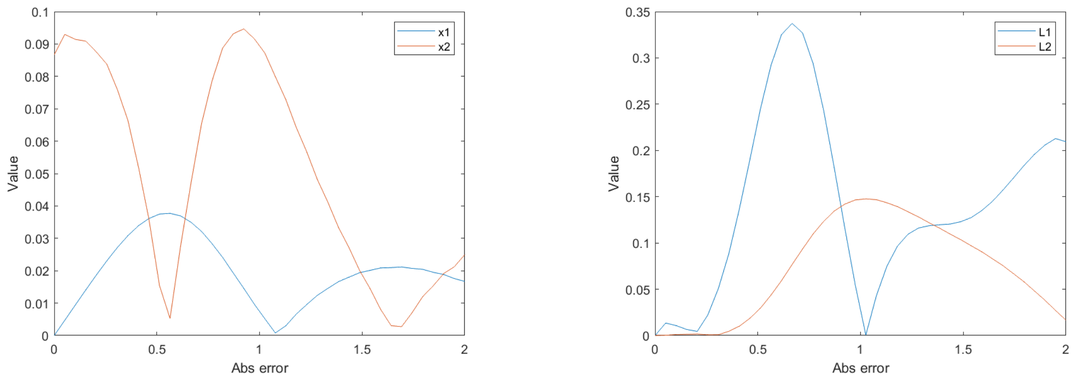


Figure 3.14: Absolute Error for States and Costates with Gaussian CRBF

CHAPTER 4: CONCLUSION

For this research, a user-friendly MATLAB toolbox named Radial Optimal Control Software (ROCS) is introduced to solve nonlinear optimal control problems by applying collocation techniques using Coupled Radial Basis Functions (CRBFs). The primary advantage provided by CRBFs is the combination of traditional Radial Basis Functions with a r^5 conical spline, which maintains the accuracy and robustness of the solution without significant influence from the shape parameter nor a certain nodal distribution. Unlike other commercial software tools that require code manipulation to operate, the ROCS toolbox enables both professionals and students to solve nonlinear optimal control problems using CRBF-collocation via a user-friendly interface.

The accuracy of the ROCS toolbox's method was tested using three nonlinear optimal control problems, Lambert's problem (two-body orbital transfer), Zermelo's problem, and the fixed final state duffing oscillator problem, and the results were compared to the solutions found in literature. The tool accepted user input for boundary conditions, necessary conditions, and the governing equations of motion and obtained the solution. Upon comparison to solutions found in the literature, ROCS demonstrated efficiency for the same level of accuracy. In addition, the insensitivity to the shape parameter was demonstrated with Zermelo's Problem, while different CRBF types were demonstrated with the duffing oscillator problem.

By introducing this unified MATLAB tool to solving nonlinear optimal control problems, users are able to utilize CRBFs to solve nonlinear optimal control problems, e.g., in astrodynamics and space-flight mechanics, without the need for extensive manipulation of code in existing software tools and without extensive knowledge of applying collocation and numerical solvers.

There are several areas for future work that can be explored. First, the features for ROCS toolbox can be expanded to include different types of boundary conditions. For example, with the duffing oscillator problem, additional code can be added to handle harmonic steady state, prescribed initial and final states, and fixed final state cases. Another area for future work would be to test the ROCS toolbox with additional optimal control problems. For this research, only Lambert's problem, Zermelo's problem, and the duffing oscillator problems were explored. Finally, the ROCS toolbox could be implemented through a different coding language, such as C, to explore additional functionality.

APPENDIX : INSTRUCTIONS FOR DOWNLOADING ROCS TOOLBOX

The MATLAB version of the ROCS toolbox can be dowloaded from the following GitHub repository: https://github.com/beth112760/Radial-Optimal-Control-Software-ROCS-

LIST OF REFERENCES

[1] J. Macki and A. Strauss, *Introduction to Optimal Control Theory*. Verlag, NY: Springer, 1982.

[2] L.D. Berkovitz and N.G. Medhin, *Nonlinear Optimal Control Theory*. Boca Raton, FL: CRC Press, 2013.

[3] D.E. Kirk, *Optimal Control Theory: An Introduction*. Englewood Cliffs, NJ: Prentice-Hall, 1970.

[4] A.V. Rao, ”A Survey of Numerical Methods for Optimal Control,” *A Survey of Numerical Methods for Optimal Control*, 1st ed., vol. 135, pp. 497-528

[5] D. Benson, G. Huntington, T. Thorvaldsen, and A. Rao, ”Direct Trajectory Optimization and Costate Estimation via an Orthogonal Collocation Method,” *AIAA Guidance, Navigation, and Control Conference and Exhibit,* vol. 29, no. 6, 2006.

[6] J.T. Betts, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. Philadelphia, PA: SIAM, 2010.

[7] O. von Stryk, R. Bulirisch, *Direct and indirect methods for trajectory optimization. Annals of Operation Research.* vol 37, no. 1, pp. 357-373, 1992.

[8] G. Huntington, A. Rao, *”Comparison of Global and Local Collocation Methods for Optimal Control”, Journal of Guidance, Control, and Dynamics.* vol 37, no. 2, pp. 432-436, 2008.

[9] B. Fornberg, *Practical Guide to Pseudospectral Methods.* Cambridge Univ. Press, Cambridge, England, U.K., 1998.

[10] G. Huntington, A. Rao, *"Optimal Spacecraft Formation Configuration Using a Gauss Pseudospectral Method", Proceedings of the 2005 AAS/AIAA Spaceflight Mechanics Meeting.* American Astronautical Society Paper 05-103, 2005

[11] G. Huntington, A. Rao, *"Optimal Reconfiguration of a Spacecraft Formation via a Gauss Pseudospectral Method", Proceedings of the 2005 AAS/AIAA Astrodynamics Specialist Conference.* American Astronautical Society Paper 05-338, 2005.

[12] D, Garg, M. Patterson, W. Hager, A. Rao, D. Benson, G. Huntington, *A unified framework for the numerical solution of optimal control problems using pseudospectral methods, Automatica.* vol 46, no. 11, pp. 1843-1851, 2010.

[13] G,. Elnagar, M. Kazemi, M. Razzaghi, *The pseudospectral legendre method for discretizing optimal control problems, Automatic Control, IEEE Transactions on.* vol 40, no. 10, pp. 1793-1796, 1995.

[14] F. Fahroo, I.M. Ross, *Costate estimation by a legendre pseudospectral method, Journal of Guidance, Control, and Dynamics.* vol 24, no. 2, pp. 270-277, 2001.

[15] G. Elnager, M. Kazemi, *Pseudospectral chebyshev optimal control of constrained nonlinear dynamical systems, Computational Optimization and Applications.* vol 11, no. 2, pp. 195-217, 1998.

[16] F. Fahroo, I.M. Ross, *Convergence of the costates does not imply convergence of the control. Journal of guidance, control, and dynamics.* vol 31, no. 5, pp. 1492-1497, 2008.

[17] H. Mirinejad, T. Inanc, *An RBF Collocation Method for Solving Optimal Control Problems, Robotics and Autonomous Systems.* vol 87, pp. 219-225, 2017.

[18] R. Hardy, *Multiquadric equations of topography and other irregular surfaces, J. Geophys. Res.* vol 76, no. 8, pp. 1905-1915, 1971.

[19] E.J. Kansa, *Multiquadrics- a scattered data approximation scheme with applications to computational fluid dynamics- 1 Surface approximations and partial derivative estimates, Comput. Math. Appl.* vol 19, pp. 127-145, 1990.

[20] J.A. Rad, S. Kazem, K. Parand, *Radial basis functions approach on optimal control problems: a numerical investigation. Journal of Vibration and Control.* vol 20, no. 9, pp. 1394-1416, 2014.

[21] T. Elgohary, L. Dong, J. Junkins, S. Atluri, *A simple, fast, and accurate time-integrator for strongly nonlinear dynamical systems, CMES: Computer Modeling in Engineering and Sciences.* vol 100, no. 3, pp. 249-275, 2014.

[22] T. Elgohary, L. Dong, J. Junkins, S. Atluri, *Time domain inverse problems in nonlinear systems using collocation and radial basis functions, CMES: Computer Modeling in Engineering and Sciences.* vol 100, no. 1, pp. 59-84, 2014.

[23] M. Mongillo, *Choosing basis functions and shape parameters for radial basis function methods. SIAM undergraduate research online.* vol 4, no. 190-209, pp. 2-6, 2011.

[24] A.D. Cheng, *Multiquadric and its shape parameter—a numerical investigation of error estimate, condition number, and round-off error by arbitrary precision computation. Engineering analysis with boundary elements.* vol 36, no. 2, pp. 220-239, 2012.

[25] A. Karageorghis, P. Tryfonos, *Shape parameter estimation in rbf function approximation. International Journal of Computational Methods and Experimental Measurements.* vol 7, no. 3, pp. 246-259, 2019.

[26] Y. Zhang, *An accurate and stable rbf method for solving partial differential equations, Applied Mathematics Letters.* vol 97, pp. 93-98, 2019.

[27] A. Seleit, T. Elgohary, *A Shape Parameter Insensitive CRBFs-Collocation for Solving Nonlinear Optimal Control Problems.* [Unpublished Manuscript]. Department of Mechanical and Aerospace Engineering, University of Central Florida. 2022.

[28] S. Ozana, T. Docekal, J. Nemcik, F. Krupa, J. Mozaryn, *A Comparative Survey of Software Computational Tools in the Field of Optimal Control, 2021 23rd International Conference on Process Control (PC).* 2021.

[29] J.C. Butcher, *Numerical Methods for Ordinary Differential Equations.* New York: John Wiley and Sons, 2008.

[30] L.F. Shampine, J. Jierzenka, M.W. Reichelt *Solving Boundary Value Problems for Ordinary Differential Equations in MATLAB with bvp4c.* [Online]. Available: https://classes.engineering.wustl.edu/che512/bvp_paper.pdf. [Accessed: 2-July-2022]. 2000.

[31] P. Falugi, E. Kerrigan, E. van Wyk *Imperial College London Optimal Control Software User Guide (ICLOCS).* [Online]. Available: http://www.ee.ic.ac.uk/iclocs/user_guide.pdf. [Accessed: 2-July-2022]. 2010.

[32] Y. Nie, O. Faqir, E.C. Kerrigan, *ICLOCS2: Try this optimal control problem solver before you try the rest, 2018 UKACC 12th International Conference on Control.* [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=arnumber=8516795. [Accessed: 2-July-2022]. Sheffield, UK. 2018.

[33] A. Wachter, L.T. Biegler, *On the implementation of an ¨ interior-point filter line-search algorithm for large scale nonlinear programming, Mathematical Programming.* vol 106, no. 1, pp. 2557, 2006.

[34] MathWorks, *Find minimum of constrained nonlinear multivariable function - MATLAB fmincon.* [Online]. Available: https://www.mathworks.com/help/optim/ug/fmincon.html. [Accessed: 2-July-2022]. 2017.

[35] C. Buskens, D. Wassel, *The ESA NLP Solver WORHP. In Modeling and Optimization in Space Engineering, G. Fasano and J.D. Pinter (Eds.).* vol. 3. Springer, New York. 2013.

[36] M. Patterson, A. Rao, *GPOPS-II: A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using hp-Adaptive Gaussian Quadrature Collocation Methods and Sparse Nonlinear Programming, ACM Transactions on Mathmatical Software.* vol 41, no. 1, pp. 1-37, 2014.

[37] I.M. Ross, *Enhancements to the DIDO Optimal Control Toolbox, ArXiv* [Online]. Available: https://arxiv.org/abs/2004.13112. [Accessed: 2-July-2022]. 2020.

[38] D. Ariens, M. Diehl, H. Ferreau, F. Logist, R. Quirynen, M. Vukov, *ACADO Toolkit User's Manual* [Online]. Available: http://acado.sourceforge.net/doc/pdf/acado_manual.pdf. [Accessed: 2-July-2022]. 2014.

[39] M. Cizniar, M. Fikar, M.A. Latifi, *Dynamic Optimisation CodeDYNOPT. User's Guide.* [Online]. Available: https://usermanual.wiki/Document/dynoptguide.491293101/view. [Accessed: 2-July-2022]. Technical Report, KIRP FCHPT STU Bratislava, Slovak Republic. 2006.

[40] M. Kelly, *OptimTraj Users Guide.* [Online]. Available: https://usermanual.wiki/Pdf/OptimTrajUsersGuide.958170878.pdf. [Accessed: 2-July-2022]. 2016.

[41] D. Henrion, J. Lasserre, C. Savorgnan, *POCP: a Package for Polynomial Optimal Control Problems.* [Online]. Available: https://arxiv.org/pdf/0809.4623.pdf. [Accessed: 2-July-2022]. 2009.

[42] A. Schwartz, Y. Chen, *RIOTS95-a MATLAB Toolbox for Solving General Optimal Control Problems and Its Applications to Chemical Processes.* [Online]. Available: http://www.optimization-online.org/DB_FILE/2002/11/567.pdf. [Accessed: 2-July-2022]. 2002.

[43] K. Holmstrom, A. Goran, M. Edvall, *User's Guide for TOMLAB.* [Online]. Available: https://tomopt.com/docs/TOMLAB.pdf. [Accessed: 2-July-2022]. 2010.

[44] A.E. Bryson and Y. Ho, *Applied Optimal Control: Optimization, Estimation, and Control*. New York, NY: Taylor Francis Group, 1975.

[45] F.L. Lewis, D.L. Vrabie, V.L. Syrmos, *Optimal Control*. Hoboken, NJ: Wiley, 2012.

[46] D.L. Torre, R. Flores, E. Fantino, *On the solution of Lambert's problem by regularization. Acta Astronautica* 2018.

[47] R.H. Battin, *An Introduction to the Mathematics and Methods of Astrodynamics, Revised Edition*. Reston, VA: American Institute of Aeronautics and Astronautics, Inc, 1999.

[48] H. Schaub and J.L. Junkins, *Analystical Mechanics of Space Systems*. Reston, VA: American Institute of Aeronautics and Astronautics, Inc, 2003.

[49] E. Zermelo, *Über das Navigationsproblem bei ruhender oder veränderlicher Windverteilung*. Zeitschrift für Angewandte Mathematik und Mechanik., 1931.

[50] L. Biferale, F. Bonasccorso, M. Buzzicotti, P.C. Di Leoni, and K. Gustavsson, *Zermelo's problem: Optimal point-to-point navigation in 2D turbulent flows using Reinforcement Learning*. An Interdisciplinary Journal of Nonlinear Science, 2019.

[51] Kovacic and M. J. Brennan, *The Duffing Equation: Nonlinear Oscillators and Their Behavior*. Hoboken, NJ: Wiley, 2011.

[52] A. H. Salas Salas, J. E. Castillo Hernández, L. J. Martínez Hernández, *The Duffing Oscillator Equation and Its Applications in Physics*. Mathematical Problems in Engineering, 2021.

[53] MathWorks, *ODE23, Solve nonstiff differential equations - low order method - MATLAB*. [Online]. Available: https://www.mathworks.com/help/matlab/ref/ode23.html. [Accessed: 12-June-2022].

[54] N. Bedrossian, S. Bhatt, M. Lammers and L. Nguyen, *Zero Propellant Maneuver: Flight Results for 180 ° ISS Rotation.* NASA CP Report 2007-214158; 20th International Symposium on Space Flight Dynamics, September 24-28, 2007, Annapolis, MD.